THE CONSTRUCTION OF ROUTES AND SCHEDULES FOR THE TRANSPORTATION OF THE HANDICAPPED

WORKING PAPER

by

Serge Roy, Luc Chapleau, Jacques Ferland
Guy Lapalme and Jean-Marc Rousseau

Ce rapport a été présenté au Congrès de 1983 de l'Association des routes et transports du Canada (A.R.T.C./R.T.A.C.) du 26 au 29 septembre 1983 à Edmonton, Alberta, Canada.

Université de Montréal
Centre de recherche sur les transports - Publication #308
Juin 1983

Dépôt légal - 3e trimestre 1983

ABSTRACT

A heuristic is developed for solving the multi-vehicle advance request and demand responsive problems. The objective is to maximize vehicle productivity for a fixed level of service.

The heuristic is tested on a set of data from several cities.

- - - - - - - - - - - - - -

1. INTRODUCTION

The number of handicapped persons using dedicated urban public transportation is constantly increasing. This places pressure on the service to improve both its efficiency and accessibility. While there exist management information systems to help any administration in handling the complex task of reservation, scheduling and dispatching there is at present no adequate algorithm available to generate a good schedule from a large group of travel requests. Several projects are currently being pursued in the U.S.A. and we report in this paper on the project carried by the Centre de recherche sur les transports of the Université de Montréal under a grant from the R.T.A.C. cooperative research program and Transport Canada Transportation Development Centre. The present study is thus aimed at the development and testing of advanced software for the automatic construction of vehicle routes and schedules for a semi-collective transportation service for the handicapped.

This type of transportation is expensive in terms of human and financial resources. The manual construction of routes and schedules for dozens of vehicles transporting hundreds of users is an extremely complex task. For example, the Montréal Urban Community Transit Commission (M.U.C.T.C.) has to construct routes for approximately fifty vehicles to serve an area of about 300 square kilometers.

Two types of problems related to this kind of transportation have been studied in the literature. The first problem involves the construction of routes to respond to a set of requests known in advance [2,5]. The second involves the integration of a new request into an already existing route [4]. In this paper, we propose a heuristic which can handle both problems in which the second problem merely becomes a special case of the first. This heuristic can solve large scale problems. In the first step, the objective is to maximize vehicle productivity with the quality of service fixed for each request. In the next step, the quality of service is maximized while maintaining vehicle productivity at the value obtained in the previous step.

Our heuristic is based on a parallel insertion technique [3]. For a given level of service, we specify a time interval during which each stop must be made. A measure of affinity between a request and a route is defined as a function of the time windows and the distance between the stops involved. This measure is used to guide the parallel insertion of groups of requests. Two requests are placed in the same group if they have a common stop in terms of both time and space. We demonstrate the usefulness of this approach using results on a number of different problems.

2. THE DATA

2.1 Requests

Each request involves the movement of an individual or group from

a place of origin to a destination. A time for the service is specified corresponding to a desired departure or arrival time, depending on the type of request. The group to be transported may include handicapped persons, who are ambulatory or in a wheel chair, and accompanying persons.

For each request (i) the following data is available :

o(i)	: place of origin indicator for request (i)
d(i)	: destination indicator for request (i)
NBAM(i)	: number of ambulatory handicapped persons
NBCH(i)	: number of handicapped persons in wheel chairs
NBCO(i)	: number of accompanying persons
PRIORITE(i)	: quality of service to be provided
TYPE(i)	: the type of request (departure time required or arrival time required)[1]
HS(i)	: the arrival or departure time required depending on the type of request (TYPE(i)).

2.2 Driver duty

A set of driver duty myst be input into the heuristic. These duties can be supplied by the user, or produced by HASTUS-macro, an algorithm for driver duty construction (see Blais and Rousseau [1]). The use of this algorithm requires an estimate of the number of vehicles required for each period of the day, which can be obtained by examining the average demand profile. It is important to note that the heuristic may only use a subset of the duties proposed. Each route produced by the heuristic corresponds to either a duty of a piece of work of a duty[2].

For each driver duty (j) we have the following information :
nbp(j) : the number of pieces of work in workday j.

For each piece of work k in the workday j :

o(k)	: indicator of location where piece of work k begins
d(k)	: indicator of location where piece of work k finishes
Hd(k)	: starting time for piece of work k
Hf(k)	: finishing time for piece of work k.

2.3 The distance table

Each request involves two stops : an origin and a destination. So for a set of n requests we have 2n stops, and these stops correspond to m different locations ($m \leq 2n$). The set of stops assigned to a given location depends on the location method.

2.3.1 Location methods

These location methods could be used :

---

[1] The user can either require a specific departure time or arrival time but not both.

[2] Duties for drivers in transit are in general made of one or two pieces of work.

i) All the stops with the same address are said to be at the same location.
ii) Stops are grouped together according to the distance separating them, and a specific location is associated with each group.
iii) The territory is divided into sectors which may be determined by postal codes, by a grid system or manually. The centroid of each sector determines a location.

2.3.2 Estimation methods

Let d(i,j) be the distance between location (i) and location (j). This can be evaluated in one of two ways.
A) Each location is geocoded and the Euclidean distance is calculated taking into account barriers and congestion zones [7]
B) Each location is associated with a node in a network (which may be complete or aggregated). The distance is calculated by determining the shortest path in this network.

2.3.3 Distance table

Because the estimation of the distance between two points is needed very often in the algorithm and while efficient it is a time consuming taks we have chosen like most authors in the field to work with distance table. We consider two types of tables :
A complete table: the distances between each pair of locations is known
A partial table : for a given location, the distance to neighboring locations only is known.

The type of table constructed, and the location and estimation methods used are chosen according to the number of locations, and the surface area of the territory to be covered. For m locations, the dimension of the complete table is m(m+1)/2.

d(i,j) : the distance between location (i) and location (j), j<i
d(i,i) : the error due to the method of location; the average distance between two points assigned to the same location.

2.4 The speed table

The system allows the user to define speed as a function of the distance. This approach is more realistic, as for short distances, there is always a fixed time required for acceleration and deceleration, and for long distances higher speeds are possible because of the ability to use fast lanes.

The function is specified using six parameters :
VITMIN : minimum speed (metres/minute)
VITMOY : average speed (metres/minute)
VITMAX : maximum speed (metres/minute)
d1 : the distance at which the average speed is applied (metres)

d2 : the distance at which the speed becomes higher than the average speed (metres)
d3 : the distance above which the maximum speed is applied.

Using these parameters, the following function is applied :

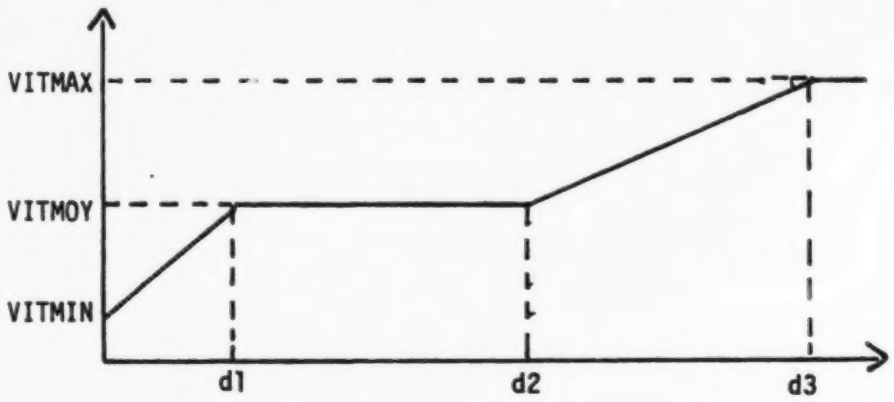


Figure 1 - Speed function

2.5 Dwell times

For each stop, a handling time is calculated. This time is a function of the number and type of handicapped included in the request corresponding to this stop. The user must supply the following values :
CHAISEMAN : time required for handling a wheelchair
AMBUMAN : time required for handling an ambulatory handicapped person.

If the request involves several people, then the complete handling time for the first individual is included plus half the basic handling time for each additional person. This method of calculation takes into account the fact that the driver has alighted and the doors of the vehicle are open.
If (i) is a stop for request "d", the handling time at stop (i) is thus given by :

$$TM(i) \begin{cases} = (nbch(d)+1)/2.0*CHAISEMAN+NBAM(d)/2*AMBUMAN & \text{if } nbch(d)>0 \\ = (nbam(d)+1)/2.0*AMBUMAN & \text{if } nbch(d)=0. \end{cases}$$

The user may also input times allowed to the driver at the beginning and end of the piece of work.

DEPGAR : time allowed to the driver for departure from the garage
ARRGAR : time allowed to the driver for arrival at the garage.

2.6 The quality of service

In constructing the routes, the heuristic fixes for each

request (i) :
HD(i) : the time at which the vehicle leaves the origin or request (i)
HA(i) : the time at which the vehicle arrives at the destination of request (i).

Between the departure and arrival of passengers for a given request, the vehicle may visit other stops to serve other request thus generating excess travelling time for the passengers of the original request.

ERT(i) : excess ride time for request (i)
$ERT(i) = HA(i)-HD(i)-t(o(i),d(i))$.

For each request (i) we have in addition the request type (departure or arrival time desired) as well as the precise time HS(i) that the service is required. We define :

DTD(i) : deviation from desired service time.

$$DTD(i) = \begin{cases} HD(i)-HS(i) & \text{if type (i) = departure required} \\ HS(i)-HA(i) & \text{if type (i) = arrival required.} \end{cases}$$

The transport company specifies the constraints on the quality of service provided to users. This quality measure will depend on the priority of the request. The quality of service provided to each priority is specified by the following parameters : the maximum deviation from the desired service time allowed and the maximum excess travelling permitted. In addition, the maximum time a passenger may spend in the vehicle can also be specified.

DM(j) : Maximum deviation from desired service time for PRIORITE(j)
1) $DTD(i) \leq DM(PRIORITE(i))$
EM(j)* : Maximum excess ride permitted for PRIORITE(j)
2) $ERT(i) \leq EM(PRIORITE(i))$
VOYAGEMAX : Maximum travel time
3) $|HD(i)-HA(i)| \leq VOYAGEMAX$

* The excess ride time could be formulated in a more complex way as a percentage of the duration of a direct trip. A minimum level of excess ride could also be defined below which excess travel is not penalized. For example, if for excess ride time, 100% of travel time is allowed with a minimum of 10 minutes, it means that for a travel time of less than 10 minutes, 10 minutes of excess ride time is allowed while for a travel time of 20 minutes a 20 minutes of excess ride time is allowed.

3. DEFINITIONS

3.1 The time window concept

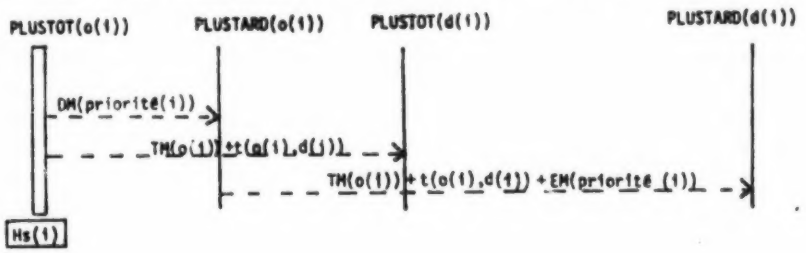
Each request involves two stops i.e. an origin and a destination. For each of these stops, we define a time window. This is the

interval of time within which the stop must be visited. These time widdows enable the constraints on the deviation from the desired service time to be satisfied, and they help restrict excess travelling. However, they do not necessarily force the constraints on maximum excess travel to be respected. When working with the time windows, the excess travel constraint is relaxed. The time windows for a request are a function of the quality of service corresponding to the priority rating of the request.

For each stop (k) we define :
PLUSTOT(k) : the earliest time the vehicle may arrive at stop (k)
PLUSTARD(k) : the latest time the vehicle may arrive at stop (k).

We now show diagramatically how the time windows are determined for the stops included in a request depending on the type of request (i.e. departure or arrival desired).

Consider a request (i) with origin o(i), destination d(i) and type TYPE (i), if TYPE (i) = departure desired



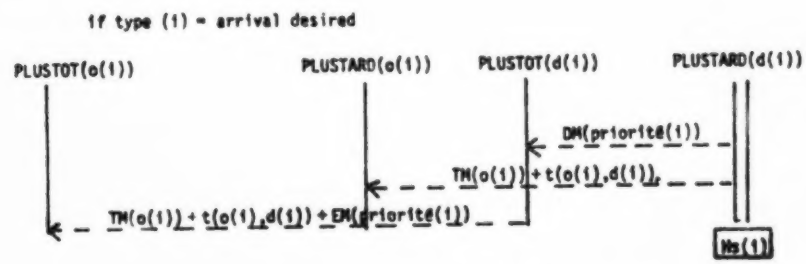


Figure 2 - Time window

By defining the time windows in this way, we respect the constraint on the maximum deviation from the desired service time. The constraints on the maximum excess travel time and on the maximum total travel time are relaxed. These excess travel constraints become as follows :

$$4)\ ERT(i) \leq \begin{cases} EM(PRIORITE(i))+DM(PRIORITE(i))-(HD(i)-HS(i)) \\ \qquad \text{if type (i) = departure required} \\ EM(PRIORITE(i))+DM(PRIORITE(i))-(HS(i)-HA(i)) \\ \qquad \text{if type (i) = arrival required} \end{cases}$$

$$5)\ |HD(i)-HA(i)| \leq \begin{cases} VOYAGEMAX+DM(PRIORITE(i))-(HD(i)-HS(i)) \\ \qquad \text{if type (i) = departure required} \\ VOYAGEMAX+DM(PRIORITE(i))-(HS(i)-HA(i)) \\ \qquad \text{if type (i) = arrival required} \end{cases}$$

The values of EM(PRIORITE(i)) and/or DM(PRIORITE(i)) could be altered in order to respect the constraints specified by the transport company. But this approach would not allow a request to be satisfied at the limits on the deviation from the desired service time and at the limit of excess travel time. This would leave little latitude to the heuristic. It is better to initially relax the constraints on excess travel and on total travelling and then to come back to them during the post optimization phase moving those request which do not satisfy the constraints.

3.2 The "neighbour" concept

In most methods proposed in the literature [2,5], the problem with several vehicles is tackled by applying a clustering technique. The multiple vehicle problem is reduced to several problems with one vehicle by determining the requests to be handled by the same vehicle. The difficulty with this approach lies in the construction of clusters. Although it is not easy to identify all the requests to be handled by a single vehicle, it is however relatively easy to see that certain requests will be handled by the same vehicle. Requests with stops which are close to one another in both space and time are likely to use the same vehicle. This led us to define the concept of a mini cluster or group of neighbouring requests.

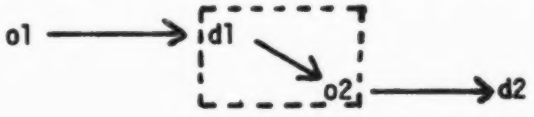
We define the constant :

VOISINMAX : the maximum distance allowed between two stops related to two different requests for the requests to be considered as "neighbours".

A necessary condition for two requests to be neighbours is that one involves a stop (ℓ) and the other a stop (k) such that

1) $\text{plustard}(k) - \text{plustot}(\ell) \geq t(\ell,k) \leq \text{VOISINMAX}$.

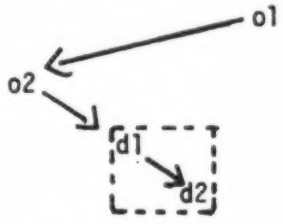
If stops (k) and (ℓ) are of a different type, i.e. one is an origin and the other a destination, then the necessary condition is also sufficient. An example of a possible case is illustrated below



d1 corresponds to stop (ℓ) and o2 corresponds to stop (k). If

condition (1) is satisfied, the requests are considered to be neighbours. The position of the other two stops is not relevant here.

If stops (ℓ) and (k) are of the same type, i.e. both are origins, or both are destinations, condition (1) is not sufficient and the position of the other two stops involved must be considered to ensure that the requests are going in the same direction. The additional condition required is illustrated by an example.



Suppose d1 and d2 satisfy condition (1). We require in addition that :

a) $t(o1,o2) + t(o2,d1) < 1.4 \cdot t(o1,d1)$.

The constant 1.4 is a parameter which approximately defines an ellipse around the stops. This value was determined from tests carried out on a school transportation problem.

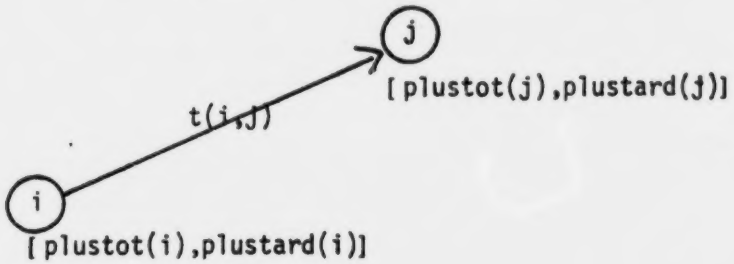
3.3 A measure of affinity

We wish to define a measure of affinity between a request and a route to guide the parallel insertion algorithm. This measure must reflect the case with which the request can be inserted into the route. The request has two stops : an origin and a destination. The precedence relationship between the origin and the destination must be respected. Each stop in the request and on the route has an associated time window. These time windows are adjusted to take into account the effect of other stops on the same route. Thus, it is easy to see that knowledge of the distances separating stops associated with a request from stops on a route is insufficient to determine this measure of affinity because of the need to consider the "time" dimension associated with the time windows. Our measure must therefore integrate both "space" and "time" dimensions.

We first introduce a method of checking whether the sequencing of any stops is feasible. The same idea is used to check the feasibility of linking two requests. For a given request and route, if there exists a feasible link between the request and another request already inserted into the route, then our measure provides an indication of the detour caused by this link. Otherwise, i.e. when there is no request already in the route which can be linked with the given request, then the measure takes a very large value indicating that the insertion of the request into the route is infeasible. Thus, the lower the value of the measure,

the greater the affinity between the route and the request and the easier will insertion be.

Let (i) and (j) be two stops with time windows: [plustot(i), plustard(i)] and [plustot(j), plustard(j)] respectively. Let t(i,j) be the time required to go from (i) to (j).



Considering these two stops, the sequence (i) → (j) is feasible if we have

$$plustard(j) - plustot(i) - t(i,j) > 0 .$$

3.3.1 The sequencing of two requests

In order to visualize the problem, refer to the following diagram. The request shown by the dotted lines is our reference request, and this is considered in relation to four other requests with their time windows.

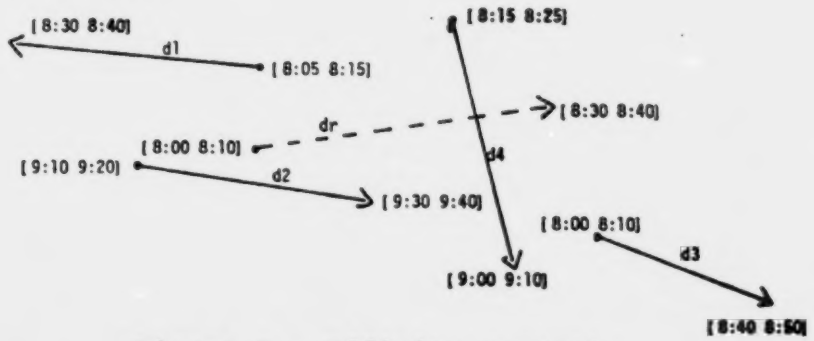


Figure 3 - Affinity request

Which of the requests d1, d2, d3 and d4 has greatest affinity with the reference request (dr)? Request d1 is in the opposite direction. Request d2 occurs too late and request d3 seems too far away. If the sequencing of dr and d4 given by o(dr)→o(d4)→ d(dr)→d(d4) is examined, it can be seen that request d4 seems to have the greatest affinity with dr.

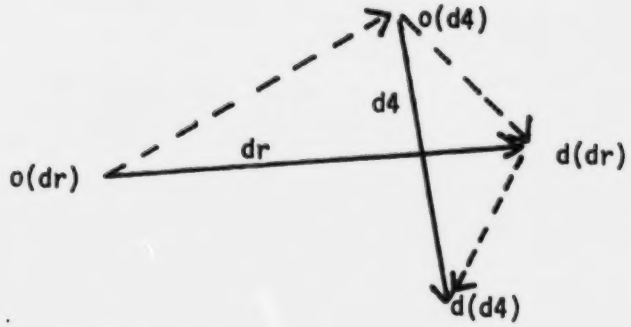


Figure 4 - Sequencing request

Note that the affinity between two requests depends on the particular sequencing considered. Given the precedence relations, there are six ways of linking two requests. For example, for requests (k) and (ℓ) we may have :

1) o(k) → d(k) → o(ℓ) → d(ℓ)
2) o(ℓ) → d(ℓ) → o(k) → d(k)
3) o(k) → o(ℓ) → d(k) → d(ℓ)
4) o(k) → o(ℓ) → d(ℓ) → d(k)
5) o(ℓ) → o(k) → d(ℓ) → d(k)
6) o(ℓ) → o(k) → d(k) → d(ℓ)

Based on the same principle used above to check on the feasibility of sequencing two stops, constraints can be defined to check on the feasibility of any given two requests.

In general, we have the sequence :

$$a \rightarrow b \rightarrow c \rightarrow d \ .$$

This involves a succession of three sequences of two stops. Each of these must be feasible for the total sequence to be valid. A sequencing of two requests is feasible if it satisfies the following three constraints :

1) The first constraint ensures the feasibility of the sequence :

$$a \rightarrow b$$

$$\text{Min } \{\text{plustard}(b), \text{Min } \{\text{plustard}(c), \text{plustard}(d) - t(c,d)\} - t(b,c)\} - \text{plustot}(a) - t(a,b) > 0.$$

2) The second ensures the feasibility of the sequence :

$$b \rightarrow c$$

$$\text{Min } \{\text{plustard}(c), \text{plustard}(d) - t(c,d)\} - \text{Max } \{\text{plustot}(b), \text{plustot}(a) + t(a,b)\} - t(b,c) > 0$$

3) The third ensures the feasibility of the sequence

$$c \rightarrow d$$

$$plustard(d) - Max\{plustot(c), Max\{plustot(b), plustot(a) + t(a,b)\} + t(b,c)\} - t(c,d) > 0.$$

For the 4th and 6th ways of linking the two requests, b → c is automatically feasible, so constraint (2) does not have to be checked and constraints (1) and (3) can be simplified to give :

1) $Min\{plustard(b), plustard(d) - t(c,d) - t(b,c)\} - plustot(a) - t(a,b) > 0.$
3) $plustard(d) - Max\{plustot(c), plustot(a) + t(a,b) + t(b,c)\} - t(c,d) > 0.$

For the 1st and 2nd ways of linking the requests, a → b and c → d are automatically feasible, so constraints (1) and (3) do not have to be checked and constraint (2) can be simplified to give :

2) $plustard(c) - plustot(b) - t(b,c) > 0.$

For the third and 5th ways of linking the requests, the constraints must be used in their most general form.

3.2.2 A measure of affinity between a request and a route

Consider a request (d) and a route (p). Using the constraints of section 3.3.1 we determine feasible links between the request (d) and the requests in the route (p). For each feasible link we estimate the detour for the route. This depends on the particular sequencing of stops considered.

For links of the 1st and 2nd type, which involve the movement t(b,c) of a possible empty vehicle we use as an estimate of the detour :

$$1{\cdot}5 * t(b,c) + t(c,d) .$$

This penalizes empty running.

For links of types 3, 4, 5 and 6 we use as an estimate :

$$t(a,b) + t(c,d).$$

For these sequences, t(a,b) corresponds to the time required to move from one origin to the other and t(c,d) is the time to go from one destination to another.

We define a matrix with elements :

INSERT(d,p) : the measure of affinity between request (d) and route (p).

$$INSERT(d,p) = \begin{cases} \text{minimum detour over all feasible links between request (d) and the requests of route (p)} \\ \quad \text{- if there is at least one feasible link} \\ \text{MAXMESURE} \\ \quad \text{- if there are no feasible links} \end{cases}$$

Thus, INSERT(d,p) indicates either that insertion of the request (d) into the route (p) is infeasible, or that the insertion is

feasible and in this case, the function also provides an estimate of the detour involved in inserting request (d) into route (p). With this measure, it is now possible to estimate the cost of an insertion by considering the detour involved and the cost of using the vehicle on the route considered.

It is important to note that the value of INSERT(d,p) can change if a new request (j) is inserted into route (p). For, this insertion may modify the time windows of the stops on route (p). After each insertion of a request into a route, the time windows of the stops are adjusted. If for a stop k on a route we have a time window [plustot(k),plustard(k)], we obtain a new time window [plustot*(k),plustard*(k)] after inserting request (j) into the route. Plustot*(k) will be the earliest time at which it is possible to arrive at stop (k) given the time windows on stops preceding (k). Plustard*(k) is the latest time at which the vehicle may arrive at stop (k) to be able to respect the time windows of the stops subsequent to stop (k). After insertion of a new request into the route we obtain :

$$\text{plustot*}(k) \geq \text{plustot}(k)$$
$$\text{plustard*}(k) \leq \text{plustard}(k)$$

3.4 The horizon concept

In the TAXI system ("A realistic advance reservation dial-a-ride algorithm" [6]) the requests which are known in advance and which are used to construct the routes, are classified according to the potential difficulty of inserting them into a route. They are them inserted one at a time in order of classification.

The order of insertion of requests is independent of the routes under construction. With our approach, we have a measure which reflects the difficulty of inserting a request into a route and which takes into account the location of the routes. We use this measure to determine the order of insertion of the requests. As an afternoon does not influence the insertion of a morning request, it is useless to consider all the requests for a given day at once. We therefore consider at the same time only requests having their desired service time within a given horizon. This horizon is determined by two values :

INFHORIZON : the starting time of the horizon corresponds to the earliest desired service time amongst all requests not yet inserted.

SUPHORIZON : the ending time of the horizon corresponds to the desired service time of the next request to enter into the horizon.

The management of the horizon is based on several values :

DEMHORIZON : the number of requests presently in the horizon.
HORIZONMIN : the minimum time interval allowed for the horizon.
MAXDEMHOR : the maximum number of requests allowed within the horizon.

The horizon moves forward in time by adding the next request as long as

(SUPHORIZON-INFHORIZON) < HORIZONMIN and DEMHORIZON < MAXDEMHOR.

For all requests within the horizon we have measures of affinity with each request in the routes. These measures will determine the order of insertion of the requests. This continual adjustment of the horizon ensures that no requests are left behind without being inserted.

4. THE INSERTION HEURISTIC

The heuristic is based on a parallel insertion technique. All routes are constructed at the same time. After the initialization of a few routes, requests are inserted into them, or as needed new routes can be initialized.

There are two ways of viewing the problem : for a given request within the horizon, the best route must be found to receive it; this is what we call "phase 1" of insertion. Also, for a given route, the best request must be found to be inserted into it; this we call "phase 2" of insertion. The global stratevy involves alternating these two phases.

4.1 Insertion of a request into a route

A request is to be inserted into a route. An insertion is defined by a pair of stops (GORI, GDES).

GORI : the stop in the route which precedes the point where the origin of the request is to be inserted.

GDES : the stop in the route which precedes the point where the destination of the request is to be inserted (If GDES = GORI, then the destination of the request is inserted immediately after its origin.

A pair (GORI, GDES) with GORI ≤ GDES, forms a feasible insertion when none of the time windows of the stops in the route after the insertion close up completely when they are recalculated. We still have :

$$PLUSTOT^*(k) \leq Plustard^*(k) \quad \text{for all stops } k \text{ of the route.}$$

Each feasible insertion involves a corresponding detour of the route and some inconvenience for the customer. A feasible insertion may be chosen according to different criteria :

1) that which involves the minimun detour;
2) that which causes the least inconvenience.

4.2 Initialization of a route

If, at a given step, it is impossible to find a route for a request within the horizon, a new route is initialized. The initialization of a new route involves finding an unused driver

piece of work which may include the request. The piece of work is chosen so that the time between the starting time of the piece and the time at which the vehicle must respond to the call is as small as possible, while respecting the following priority order:

1) Pieces of work from duties already begun (Days for which a piece of work has already been used).
2) First pieces of work in workdays not yet started.
3) Other pieces of work.

4.3 The phase of insertion

Phase 1

A request "d" is to be inserted. Phase 1 involves finding the best route to receive request "d". Routes are evaluated in increasing order of the measure INSERT(d,p). The best route for "d" is p* suth that INSERT(d,p*) = Min {INSERT(d,p)} ∀ p.

Phase 1 exchange

This plays the same role as phase 1, except that before attempting to insert request (d) into a route (p), one request is removed from p and relocated in another route. The request removed is d* such that d* has the best sequencing affinity with (d). Phase 1 exchange is used at the postoptimization stage.

An attempt is made to insert (d) in P only if a new route has been found for d*.

Phase 2

A request is to be inserted into route (p). Phase 2 of the insertion involves finding the best request which can be inserted into route (p). The best request for route (p) is d* such that INSERT (d*,p) =

Min {INSERT(d,p)} ∀ requests (d) within the horizon

4.4 The heuristic

1) Sort the requests into desired service time order.
2) Read the requests to initialize the horizon and to construct groups of neighbouring requests.
3) Initialize one or several routes with groups of neighbouring requests.
4) WHILE (all requests not inserted)

BEGIN

4.1 - IF (SUPHORIZON-INFHORIZON < HORIZONMIN and DEMHORIZON < MAXDEMHOR
Then
BEGIN
- Increase the horizon by reading requests.
- For each request read. initialize a row in the INSERT matrix.

- Construct and modify the groups of neighbouring requests.

END

4.2 - Determine the "worst" request d*.

4.3 - Attempt a PHASE 1 with d*

4.4 - IF (d* is not inserted) THEN relax the time windows of d* and again try PHASE 1

4.5 - IF (d* is not inserted)

THEN

BEGIN

- Initialize a route for d*
- Initialize a column in the INSERT matrix for the new route.

END

4.6 - Attempt to insert the requests neighbouring d" into the same route.

4.7 - Do a PHASE 2 with this route

END

5. POST-OPTIMIZATION

In this step, we attempt to minimize customer inconvenience (i.e. maximize the quality of service) while retaining the productivity level established in the previous step. For a given request (i), inconvenience is evaluated as a function of the deviation from the desired service time DTD(i), and of the excess ride time ERT(i). The user must indicate the relative importance of DTD(i) and ERT(i) using parameters associated with request priority.

For each request priority level, the user specifies :

PDTD(k) : the weight placed on deviations from the desired service time in the inconvenience function for a request of PRIORITE(k).

PERT(k) : the weight placed on excess ride time in the inconvenience function for a request of PRIORITE(k).

With these values, we define :

INCONV(i) : the inconvenience of request (i).

INCONV(i) = PDTD(PRIORITE(i))*DTD(i) + PERT(PRIORITE(i))*ERT(i).

By exchanging requests between routes using PHASE 1 EXCHANGE, we attempt to reduce first the greatest inconvenience, then the second greatest inconvenience, and so on.

6. OTHER IMPROVEMENTS

An option will be included such that standard recurrent vehicle route be defined. Those trips could correspond to nearly fixed route that a transit organization want to provide on a regular basis. Those are, for example, a route from the North end of the city to the General Hospital for appointments at 9:30 a.m. or a trip to a shopping center on Monday, Wednesday, Friday afternoon.

Those routes are defined both graphically and timewise. Requests that could be included in those routes could be treated first and given priority at the time of route construction. This process would give an organization the opportunity to give priority to certain corridors of services that are frequently used.

Another improvement that will be included is the consideration of the different costs of different types of vehicles that could be initialized to give the services : for example, van services versus taxi services. The heuristic algorithm should take into account both the variable and fixed cost in using each type of vehicles. This option will be particularly useful when taxi could be used.

SOME RESULTS

Tables 1-5 describe briefly the results obtained to date on the problems of Montréal and Sherbrooke. Further tests will be conducted on the problems of Toronto and Ottawa. These results are quite encouraging and demonstrate that better quality services could be provided with a significant reduction of costs. In other tests, mainly on the Toronto problem, the utilization of taxi will be evaluated and we expect even better results.

SHERBROOKE

Problem 1 : Construction of routes and schedules for subscribers requests.

Problem characteristics

The number of given routes	: 0
The number of requests making up given routes	: 0
The number of requests to insert	: 57
The number of passengers to insert	: 61
The number of drivers duties available	: 6
The number of possible routes	: 12

Route statistics

The number of driver duties used	: 5
The number of routes used	: 9
Total vehicle movement time	: 14h. 21 m.
Total vehicle movement time without passengers (1)	: 5h. 58 m.
Total dwell time	: 5h. 12 m.
Mean productivity level for the drivers (2)	: 1.31 pass./h.p.
Mean productivity level for the vehicles (3)	: 2.95 pass./h.v.m.

Passenger inconvenience statistics

Mean excess ride time	: 10 m.
Maximum excess ride time	: 44 m.
Mean deviation from requested time	: 12 m.
Maximum deviation from requested time	: 37 m.
Execution time	: 55.9 s.

(1) This time includes the time to travel from the garage to the first stop and the time from the last stop to the garage.

(2) This is defined as the number of passengers transported per hour paid. (There may be more hours paid than hours of vehicle movement.)

(3) This is defined as the number of passengers transported per hour of vehicle movement.

SHERBROOKE

Problem 2 : Construction of routes and schedules with telephoned requests known in advance using the routes generated for subscribers (Problem 1)

Problem characteristics

The number of given routes	: 9
The number of requests making up given routes	: 57
The number or requests to insert	: 29
The number of passengers to insert	: 36
The number of drivers duties available	: 1
The number of possible routes	: 3

Route statistics

The number of driver duties used	: 0 total: 5(6)*
The number of routes used	: 1 total: 10(12)
Total vehicle movement time	: 19h. 18 m.
Total vehicle movement time without passengers (1)	: 7h. 40 m.
Total dwell time	: 7h. 46 m.
Mean productivity level for the drivers (2)	: 2.08 pass./h.p.
Mean productivity level for the vehicles (3)	: 3.62 pass./h.v.m.

Passenger inconvenience statistics

Mean excess ride time	: 10 m.
Maximum excess ride time	: 50 m.
Mean deviation from requested time	: 11 m. (15)
Maximum deviation from requested time	: 41 m. (60)

Execution time : 83.78 s.

---

* Their results

MONTREAL

Problem 3 : Construction of routes and schedules for subscribers requests before 09:00

Problem characteristics

The number of given routes	:	0
The number of requests making up given routes	:	0
The number of requests to insert	:	127
The number of passengers to insert	:	132
The number of drivers duties available	:	62
The number of possible routes	:	75

Routes statistics

The number of driver duties used	:	31 (31)*
The number of routes used	:	31 (31)
Total vehicle movement time	:	50h. 56 m.
Total vehicle movement time without passengers (1)	:	24h. 11 m.
Total dwell time	:	11h. 12 m.
Mean productivity level for the drivers (2)	:	0.53 pass./h.p.
Mean productivity level for the vehicles (3)	:	2.13 pass./h.v.m.

Passenger inconvenience statistics

Mean excess ride time	:	9 m.
Maximum excess ride time	:	39 m.
Mean deviation from requested time	:	12 m. (23)
Maximum deviation from requested time	:	36 m. (60
Execution time	:	77 s.

* Their results

MONTREAL

Problem 4 : Construction of routes and schedules for subscribers requests after 09:00 and with telephoned requests known in advance using the routes generated for subscribers request before 09:00 (Problem 3).

Problem characteristics

The number of given routes	: 31
The number of requests making up given routes	: 127
The number of requests to insert	: 451
The number of passengers to insert	: 540
The number of drivers duties available	: 31
The number of possible routes	: 44

Routes statistics

The number of driver duties used	: 25 total:56(62)*
The number of routes used	: 35 total:65(75)
Total vehicle movement time	: 230h. 48m.
Total vehicle movement time without passengers (1)	: 87h. 19m.
Total dwell time	: 52h. 36m.
Mean productivity level for the drivers (2)	: 1.5. pass./h.p.
Mean productivity level for the vehicles (3)	: 2.3 pass./h.v.m.

Passenger inconvenience statistics

Mean excess ride time	: 11 m.
Maximum excess ride time	: 54 m.
Mean deviation from requested time	: 11 m.
Maximum deviation from requested time	: 37 m.

Execution time : 472.3 s

*Their results

MONTREAL

Problem 5 : Construction of routes and schedules with requests known in advance (subscribers and telephoned requests) using improved driver duties

Problem characteristics

The number of given routes	: 0
The number of requests making up given routes	: 0
The number of requests to insert	: 578
The number of passengers to insert	: 672
The number of drivers duties available	: 48
The number of possible routes	: 69

Routes statistics

The number of driver duties used	: 48 (62)*
The number of routes used	: 69 (75)*
Total vehicle movement time	: 233 h. 25 m.
Total vehicle movement time without passengers (1)	: 90 h. 25 m.
Total dwell time	: 51 h. 50 m.
Mean productivity level for the drivers (2)	: 1.75 pass./h.p.
Mean productivity level for the vehicles (3)	: 2.3 pass./h.v.m.

Passenger inconvenience statistics

Mean excess ride time	: 11 m.
Maximum excess ride time	: 57 m.
Mean deviation from requested time	: 13 m.
Maximum deviation from requested time	: 38 m.

Execution time : 581.3 s.

---

* Their results

CONCLUSION

We have briefly described the algorithm process that we are following for the construction of route and schedule of vehicle for the transportation of handicapped persons and some preliminary results. The algorithm will be extensively tested on the data of four transit organizations : Montréal, Toronto, Sherbrooke and Ottawa. Following these tests, we hope that this algorithm will be tested in an operational environment to produce dayly the vehicle routes and schedules.

REFERENCES

[1] BLAIS, J.-Y., ROUSSEAU, J.-M. HASTUS : An Evaluation Model for Drivers' Union Contract Negotiation in Transit Companies. Centre de recherche sur les transports, Université de Montréal, Publication #163, March 1980.

[2] BODIN, L.D., SEXTON, T.R. The Multi-Vehicle Subscriber Dial-A-Ride Problem. Working paper MS/S 83-009, University of Maryland. February 1983.

[3] CHAPLEAU, L., FERLAND, J., LAPALME, G., ROUSSEAU, J.-M. A Parallel Insert Method for the Capacitated Arc Routing Problem. Centre de recherche sur les transports, Université de Montréal, Publication #234. April 1982.

[4] HAINES, G.H., Jr., WOLF, R.N. Alternative Approaches to Demand Responsive Scheduling Algorithms. Transportation Research A, Vol. 16A, No. 1, pp. 43-54 (1982).

[5] JAW, J.J., ODONI, A.R., PSARAFTIS, H.N., WILSON, N.H.W. A Heuristic Algorithm for the Multi-Vehicle Many to Many Advance-Request Dial-A-Ride Problem. Working paper MIT-UMTA -82-3. Massachusetts Institute of Technology. June 1982.

[6] KROLAK, P.D. Taxi -A Realistic Advance Reservation Dial-A-Ride Algorithm. Internal TSD Working paper report No. DOT-TSC-UM262-PM-82-32. October 1982.

[7] VACCARO, H. Alternative Technique for Modeling Travel Distance. Masters Thesis in Civil Engineering, Massachusetts Institute of Technology. June 1974.

[8] RIVARD, G. Construction des parcours des véhicules et des horaires des chauffeurs pour le transport des personnes handicapées. Centre de recherche sur les transports, Université de Montréal, Publication #240. Mai 1981.